Statistical Study of DRAM Failure Characteristics

Andy A. Hwang
University of Toronto
hwang@cs.toronto.edu

Abstract

We present a study of memory errors pooling data from several large-scale production systems. We analyzed correctable and uncorrectable errors and their temporal and spatial distributions. To the best of our knowledge, this is the first large-scale study of production systems with physical location of the error within chips. We found that error rates are higher than traditional assumptions and exhibit high degree of spacial locality. In a commodity cluster up to 10% of nodes can be affected by uncorrectable errors annually, and up to 15% of total lost compute time can be attributed to memory-related node outages. We confirmed that machines from similar hardware platforms suffer the same aging trends. For correctable errors, we measured FIT values at 85000, which is much higher than previous studies. We discovered that errors are highly localized on a small number of nodes and chips. We also observed degradation from less to more severe errors, showing that 40% of single symbol correctable errors worsen to double symbol errors over time. We offer evidence to suggest such occurrences are due to hardware errors being the dominate failure mode, such as a bad row in the DRAM cell, whereas most previous work focuses only on soft errors.

1 Introduction and Motivation

The memory system is an important part in any computer architecture. Specifically, Dynamic Random Access Memory (DRAM) devices can be found in almost any platform from mobile devices to data centres, yet their reliability characteristics are not well understood.

DRAM is commonly sold as Dual-Inline Memory Modules (DIMMs) that can be plugged into commodity machines. Memory DIMMs are one of the most common components to fail in modern day data centres [33]. When errors happen they threaten the stability and correctness of the entire system. This has been a topic of major concern and much research has been dedicated to methods of protection from memory errors [14, 18, 20, 41]. Memory controllers are almost always equipped with error correcting codes (ECC) which has limited ability to rectify some errors. Depending on the recoverability by the ECC hardware, errors can be classified into *correctable* and *uncorrectable* errors (CE and UE).

Memory errors can have serious impacts on the operation of data centres. Uncorrectable errors can lead to machine crash, shutdown, or pollute the data being stored in memory. The details of error characteristics are not well understood. As a result, system administrators typically rely on "rules of thumb" for DIMM replacements such as one UE or an arbitrary CE threshold. The operator and material cost, plus the cost of downtime is a huge expenditure for data centres housing tens of thousands of nodes.

Previous studies on memory errors were typically conducted in lab environments, or from small-scale data collections [23, 25]. Their assumptions are often not suited to real systems and thus the applicability of these results to production environments are uncertain. Better understanding of memory errors from real-world data could help the design of hardware and software techniques and further research in this area.

Some questions we would like to answer are: How common are memory errors? How are they distributed over time and across the error hardware? What is the likely cause of these errors? These questions are important because the characteristics of errors influence the design of future memory systems, and can impact administration policies on current systems. Answering these questions would allow us to identify problems in current systems and increase the reliability of systems in general.

In this work, we focus on analyzing several datasets from real-world large-scale systems. We look at the frequencies of uncorrectable errors, node outage and lost compute time. We also investigate the distribution of

correctable errors in time and across error hardware. Using error address information, we classify error instances into different categories and provide evidence that hardware failures are a significant source of errors.

The remainder of this paper is organized as follows: Section 2 presents the background and related work on memory errors, Section 3 describes the systems and the types of logs we studied, Section 4 details the characteristics of uncorrectable errors, Section 5 documents the distribution of correctable errors, and Section 6 presents the behaviour of failed devices. We conclude and outline directions of future work in Section 7.

2 Background and Related Work

In this section we provide some background on memory errors and their mitigation. We summarize the conventional error categories, correction techniques, and previous work analyzing failure data. The effectiveness of the hardware and software used to deal with memory errors is dependent on their characteristics, which is the underlying motivation for our work.

2.1 Sources of Memory Errors

There are many potential sources for memory errors. It has been long understood that alpha particles in the operating environment can penetrate chips and significantly alter the charge of a single storage cell to causes a "bit flip" [26]. Traditionally these errors have been classified as *soft* errors since their occurrences tend to be isolated and random with no lasting effects. Alternatively, physical defects can also cause errors on DRAM chips. Defects may be introduced during fabrication or through material degradation over the lifetime of the chip. Weakness in the silicon layers can cause leakage and failure in the circuitry of the cells. Since the defect resides in hardware, it is likely that these errors will manifest repeatedly. The reoccurring errors have typically been named *hard* errors. However, some hard errors may be more appropriately termed *intermittent* errors since the manifestation is not always guaranteed [12].

The majority of previous work focuses on the assumption that soft errors are the dominate form of errors in memory systems. However, we offer evidence to suggest that hard errors contribute a significant amount of errors in large-scale systems.

2.2 Error Detection and Correction

As a result of discovering errors in the memory subsystem, algorithms have been devised to recover data that have been corrupted. The basis for error correction codes (ECC) is the XOR logic operation. The most basic encodings utilize XORs to compute the parity (number of 0s and 1s) of a given byte, and that information allows detection of single bit flips. Building on the theory of finite fields, more complicated encoding schemes were developed that corrects single-bit and detects double-bit errors (SEC/DED). Similar ECC techniques are employed in different levels of cache and throughout the memory hierarchy.

Recently, more complex schemes were proposed to provide stronger correction capabilities. IBM offered the "Chipkill" algorithm to tolerate failures of an entire DRAM chip [14]. The bits from DRAM chips are organized into the error correction symbols such that the failure of an entire chip is contained and recoverable. Some systems are designed with spare bits so they can be used instead of the failed chips (Redundant Bit Steering). More details on these techniques can be found in the literature [20].

Based on the ECC employed by the system, errors can be classified into *correctable* and *uncorrectable* errors (CE and UE). Correctable errors are silently handled by the memory controller. The error bits are recovered using ECC and the memory request is fulfilled. However, even though the system can continue to execute, if the ECC logic is required for many memory accesses it is possible to perceive decreased performance. Alternatively, if ECC is unable to correct the error, the system raises an exception and execution is typically halted. Uncorrectable errors lead to machine crashes or shutdowns, and are a major concern for large-scale systems. Note that the nature of the error (hard/soft/intermittent) is orthogonal to its recoverability.

Redundancy via multiple DIMMs have also been suggested to increase reliability. Spare DIMMs are included in the system to be used when the main DIMMs break down [18]. "Mirroring" techniques are also offered which keep synchronized copies of entire DIMMs in the system [22]. These approaches are similar to basic hard drive RAID systems and suffer the same shortcomings. Example trade-offs are lower memory capacity and increased operating costs from redundant components.

Modern systems have varying degrees of ECC capabilities. Personal desktops and laptops often have no ECC hardware at all. Commodity servers usually have basic ECC built-in (SEC/DED), and better protection such as mirroring is offered for higher-end servers. Complex algorithms such as Chipkill are usually reserved for systems where reliability is a major concern or for specialized hardware such as Blue Gene.

Researchers and hardware manufacturers have continuously proposed new concepts to tolerate failures and increase the reliability of memory systems. Intel recently began offering the "hwpoison" mechanism to mark bad

memory pages to allow attempts at recovering the running system [7]. Other approaches allow isolation of error pages or strengthen the encoding for selected parts of memory, but require special hardware or target different system architectures [21, 41].

2.3 Proactive Mechanisms

To reduce the chances of errors occurring in deployed systems, memory tests have been devised as a proactive way to verify the correctness of components. Memory components are exercised by a large number or reads and writes. Special access patterns are designed to expose potential weaknesses in the silicon. Random patterns are also used for additional test case coverage.

The memory DIMM is tested multiple times throughout its life cycle. Manufacturers actively test their products prior to shipping. Specialized machinery is used to test DRAM chips and sort them into different product grades. There are also testers for assembled DIMMs. However, these testers exercise the device under test independently and not while the DIMM is plugged into the system. Data centres typically employ a "burn-in" process for new systems, in which the system is stressed to filter out faulty components before commercial deployment.

Specialized software have been written to facilitate memory testing without requiring the DIMMs to be removed from the main board. The popular Memtest86 software is commonly used to test for memory errors [11]. These testers aim to cover the entire memory address space, and as such require the machine to be booted into their environment.

Software can also be written to test memory in user space. A program can try to allocate and exercise large amounts of memory to test for errors [30]. However, this approach is subjected to the memory allocation limits and policies of the operating system.

Many chipsets support memory "scrubbing", which is a background hardware check of memory contents. The typical scrubbing pace is slow to avoid performance impacts. The scrubbing parameters and functionalities are largely dictated by hardware vendors [6].

2.4 OS Mitigation Mechanisms

Operating systems have different mechanisms for monitoring and isolation of potential bad memory pages. The Linux Error Detection and Correction (EDAC) module displays ECC summary information from supported memory controllers [38]. Using data gathered from EDAC, administrators can decide whether to remove pages in memory to avoid faulty addresses. The Linux BadMEM patches and the "memmap=" boot-time argument allow memory page isolation or truncate the memory region available to the system [2, 31]. The Solaris operating system can "retire" memory pages that it deems faulty and migrate the contents to other healthy pages [37]. AIX offers "dynamic reconfiguration" of memory which allows for addition or removal of large chunks of memory, which may also be used to avoid memory areas with hardware faults. Online memory hotplugging for Linux is also under development, which also allows removals of large sections of memory [5].

In virtualized environments, several memory management techniques such as ballooning and hotplug, allow the hypervisor to limit the memory capacities of the guest machines [32, 39]. It is possible to use these mechanisms as a method to mitigate memory errors. Recent efforts have been made targeting guest operating systems by using page replacement or live virtual machine migration [15]. Nevertheless, these approaches are typically not applicable to the hypervisor (Domain 0) or the kernel and thus cannot offer protection for these areas.

2.5 Failure Data Analysis

Several previous studies of DRAM failure data have been published [23, 25, 24]. However, the data source are limited to a small number of production servers or controlled lab environments. Schroeder et al. analyzed data from a large fleet of production machines, but unfortunately only contains per DIMM error counts, and not location information such as the error address [34]. Therefore the characteristics of the errors within chips are not available.

Haque and Pande published a large-scale study on memory used by graphics processing units (GPUs) [17]. Although the application is different, the basic DRAM hardware is identical. The authors conclude that GPUs do exhibit memory errors and manufacturers should include ECC capabilities on graphics cards.

There is a large body of work on reliability of systems in general. Supercomputer logs have been previously studied to better understand the internal workings of these systems [27]. Failures in these systems have been investigated, though in a more general sense [33]. Much work has focused on the reliability of storage systems [9, 8, 29, 33]. Reliability of internet services have also been studied [28]. This work complements existing research by providing a detailed look at failures related to the memory components with location information.

Significant work has been done on fault injection and sensitivity analysis for OS and programs in execution [10, 13, 36, 35, 40]. For example, Yim et al. examined different objects in memory and recovery methods [40]. David and Campbell suggest several techniques for

OS recovery [13]. This paper focuses on the causes of these events, rather than the manifestation which are the errors in execution.

3 Data Sources

In this section we briefly describe the architecture of the systems we investigated, and the types of data we obtained from each source. Due to the varying nature of the logs, we made different assumptions for their analysis. We describe our methods to identify memory errors from these different logs.

3.1 Los Alamos National Lab

Our first data source is Los Alamos National Lab (LANL). LANL has made failure data from their systems publicly available [3]. The failure data contains information for over 20 high-performance computing (HPC) systems spanning nine years. Most of these systems are clusters of commodity systems (symmetric multiprocessing or SMP) with regular ECC. Selected systems with Non-Uniform Memory Access (NUMA) architectures are also present. The detailed system configurations are described in Table 1.

The logs we obtained include node outage data for all systems from 1996 to 2005. While all failures that led to node outages are recorded in the LANL logs, we were able to identify memory-related failures since the root cause of each node outage is included. Using the root cause information, we can make the assumption that uncorrectable errors occurred on a node with memory-related outages. We attribute each outage caused by DRAM as one UE on the node. It is possible that we are underestimating since not all UEs will cause node outages. We also obtained job logs from one of the systems to estimate the job interruptions caused by memory errors.

3.2 Argonne National Lab

The second data source is Argonne National Lab (ANL). We obtained data collected from their IBM Blue Gene/P (BG/P) system. The BG/P system totals 40,960 nodes, each with an identical 2 GB of memory. These customized nodes have 40 memory chips directly soldered on to the board instead of commodity memory DIMMs [1]. This design requires replacement of the entire node when DRAM chips or other on-board components require replacement.

The BG/P system implements several advanced ECC features that are designed to increase the reliability of the memory system. Extra bits are included to act as spares and parity address bits, and Chipkill is employed to tolerate the failure of an entire DRAM chip [19].

The BG/P system has an extensive reliability, availability, and serviceability (RAS) logging infrastructure. We obtained RAS logs for roughly one year. This system records instances of correctable and uncorrectable errors.

The BG/P RAS logs specifically record information about UEs in several categories. In addition to the direct reporting of UEs, the messages for machine check exceptions raised by the DDR controller were also treated as sources of uncorrectable errors.

CEs are reported at the granularity of individual jobs since logging all occurrences may lead to unacceptable overheads. At the end of each job, the total number of correctable errors encountered during execution is reported for each node (capped at 65,535). We use this information to study the distribution of correctable errors over time and at the node level.

Additionally, the first instance of correctable errors on an address is reported immediately during job execution, and during the regular diagnostics check executed on the system. The immediate reports, although an underestimation of the total errors, contains the information of the faulting address and the originating DRAM chip. We use this information to study distributions at the chip level and within error chips.

3.3 SciNet Consortium

Our third data source is SciNet, one of the largest clusters in Canada that provides computing facilities available to researchers at University of Toronto and affiliated hospitals [4]. The General Purpose Cluster (GPC) at SciNet contains 3,780 nodes, each with 16 GB of memory.

We obtained parts replacement data from this system which is manually entered by an administrator when broken hardware is replaced. The replacement log we obtained spans approximately one year. In conversation with the administrators, the standard practice is to replace DIMMs when they have had an UE (or unusually high number of CEs). We assume that each memory DIMM replacement was a result of at least one uncorrectable memory error. These numbers are believed to be relatively accurate (if not underestimating) to the number of UEs in the systems.

At present we are actively working with SciNet to further monitor their system for memory errors. We hope to include the results in future studies.

4 Uncorrectable Errors

We begin by studying the characteristics of uncorrectable errors since it is the most severe category of memory er-

Table 1: Node outage due to memory errors and FIT in LANL, ANL, and SciNet systems, LANL systems are numbered similar to the CMU paper [33]. Most LANL systems have between 3% and 10% nodes affected annually, with the exception of some platforms (G/H). The Argonne BG/P have significantly less nodes affected by UEs. However, the error rate per MB per year is relatively similar between systems.

System Overviews						Statistics				
Source	System	HW Type	Prod Date	Nodes	Node Memory (GB)	Obser. Time (Years)	# UEs	Affected Nodes (Total)	Affected Nodes (/Yr)	# UEs /MB /Yr
LANL	4	D	Apr 01	164	1 GB	4.4	51	19	3%	6.84E-05
	5	E	Sep 01	512	16 GB	3.8	386	179	9%	1.22E-05
	6	E	Dec 01	128	16 GB	0.3	5	4	10%	7.29E-06
	7	E	May 02	1,024	16 GB	3.4	567	316	9%	1.01E-05
	8	E	Oct 02	1,024	16 GB	2.9	204	127	4%	4.13E-06
	9	E	Sep 03	128	4 GB	2.0	10	10	4%	9.42E-06
	10	E	Sep 03	128	4 GB	2.0	14	11	4%	1.32E-05
	11	E	Sep 03	128	4 GB	2.0	19	16	6%	1.79E-05
	12	E	Sep 03	32	16 GB	2.0	3	3	5%	2.83E-06
	13	F	Sep 03	128	4 GB	2.0	16	11	4%	1.51E-05
	14	F	Sep 03	256	4 GB	2.0	64	39	8%	3.01E-05
	15	F	Sep 03	256	4 GB	2.0	70	45	9%	3.30E-05
	16	F	Sep 03	256	4 GB	2.0	66	38	7%	3.11E-05
	17	F	Sep 03	256	4 GB	2.0	51	39	8%	2.40E-05
	18	F	Sep 03	512	4 GB	2.0	70	36	3%	1.65E-05
	19	G	Dec 96	16	32 GB	5.8	159	16	17%	5.23E-05
	20	G	Jan 97	49	128 GB	8.7	1,080	48	11%	1.94E-05
	21	G	Oct 98	5	128 GB	6.9	52	5	14%	1.14E-05
	22	H	Nov 04	1	1,024 GB	0.9	24	1	100%	2.67E-05
ANL	Intrepid	BG/P	Nov 07	40,960	2 GB	1.0	150[1]	32	0.078%	1.79E-6
SciNet	GPC	IBM	Jun 09	3,780	2 GB	1.5	156[2]	51	0.9[3]%	2.20E-5

[1]Some nodes have repeated UEs, later analysis only counts one UE per node
[2]156 replacements, but only 63 entries contains node information
[3]Replacement dates, may not reflect error date

rors. When errors occur in memory, program code and/or data is corrupted. If the errors are not corrected, it is highly possible to have the errors taint the execution of applications or even the kernel. We analyzed the logs described in Section 3 to gain insight to the characteristics of uncorrectable errors. In the remainder of this section, we show the characteristics of UEs based on the counts extracted from the node outage, raised exception and replacement data stated in Section 3.

4.1 Nodes Affected

We calculated the fraction of nodes affected and the incident rate of memory errors. Table 1 lists the nodes affected and frequencies of UEs for LANL, ANL, and SciNet systems. The left half is the overview of the system platforms and configurations, and the right half presents the node and UE statistics.

We start by looking at the LANL data. It is clear that most LANL systems have between 3% to 10% of nodes affected by memory errors annually (platforms D, E and F). The hardware platform G and H are exceptions (17%, 11%, 14%, and 100%). They are systems with NUMA architectures. The per node memory capacities and processor counts are significantly larger for these node configurations and thus the percentage of affected nodes is skewed.

The SciNet GPC has a slightly lower node error rate than LANL systems, but this may be due to the coarser granularity of data and the fact that not all UEs may lead to immediate replacement.

The ANL BG/P, on the other hand, has a significantly lower node error rate than the rest of the systems. There are two possible reasons for this. First, while BG/P contains many more nodes, this can be attributed to the usage of Chipkill ECC and other redundancy methods. Since correctable errors are well-documented in this system, it may also be possible that replacement is done well before a node develops UEs. Secondly, it is interesting to note that the node UE rates are slightly higher than

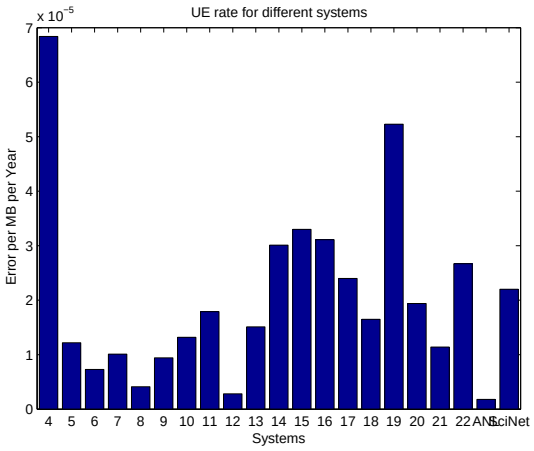

Figure 1: Error per MB per year for observed systems.

those found in [34], where the workload for the systems is not as compute and memory intensive. This could be attributed to higher memory utilization in our systems, it is possible that fewer errors were discovered in [34] simply because memory was not exercised as often.

Even though per node UE rates are different for each dataset, the error rate per MB is mostly consistent across all systems, which shows that even though affected nodes may vary, UE occurs at a similar rate for these systems. This is likely because these systems use the similar underlying DRAM technologies. Figure 1 plots the UE rate for the systems. The ANL BG/P maintains the lowest UE rate per MB per year, roughly one order of magnitude better than the worst systems, this may be attributed to stronger the ECC protection.

4.2 Hardware Platform and Aging

Next we investigate how errors are distributed over time. Understanding the distribution of errors over time may help us predict when errors are likely to occur. Figure 2 shows the cumulative distribution function (CDF) of memory-related node failure for several systems. Each plot corresponds to a different hardware platform.

We observe that there are similarities within each hardware platform. However, different platforms have different aging characteristics.

The top left plot is for hardware type E in LANL systems. While the observation period for the systems is not uniform, trends can be spotted for systems with longer observation time. Systems 5, 7 and 8 show very stable failure rate for first 15 months, then the failure rates "level off" after 18 to 25 months. The similarity within one hardware platform can be best illustrated by the top right of Figure 2, which plots the failure CDF for hardware F. All the systems in this hardware platform have a significant increase in failure rate after 13 months. The bottom left plot shows hardware group G, which has a different architecture than the rest of the LANL systems (NUMA vs SMP for other systems). The failure rates over their long observation periods are fairly steady. The bottom right plot shows the CDF of UE for ANL BG/P. While it is difficult to conclude the aging effect on this system due to the limited observation time and data points (only months 21-33 since deployment), we can see that the trend is steady within the observation period.

Table 2: Node outage to job failure matching. (Top) Using the time window criteria. Majority of matched results are within the 5 minute window. (Bottom) Using both time window and node number. Majority of matched results are also within the 5 minute window. Memory-related job failures account for over 10% of failed jobs.

Time Window (s)	Matched Count	Memory Related	Memory %
300	467	41	9%
600	592	43	7%
1,800	793	55	7%
3,600	993	58	6%
7,200	1,066	62	6%

Time Window (s)	Matched Count	Memory Related	Memory %
300	252	38	15%
600	289	41	14%
1,800	358	49	14%
3,600	388	49	13%
7,200	437	52	12%

4.3 Job Failure and Lost Compute Time

To establish a perspective on how memory errors impact the users of a system, we would like to estimate the compute time lost due to a memory-related job crash. Recall that an UE may potentially lead to node shutdown. For this purpose we obtained job logs for system 5 from LANL. These logs contain the job exit status and we matched them to node outage data. The simplest matching criterion would be the timestamps of the events. Matching was also done using a combination of timestamps and event node information. Different time windows were used and it was found that 5 minutes was a reasonable time window for matching failed jobs to node outages. Figure 3 shows the CDF of matched events vs their matched time window.

Table 2 lists the count of matched events for different criteria. Even though the total matched events increase as the time window lengthens, the total number of

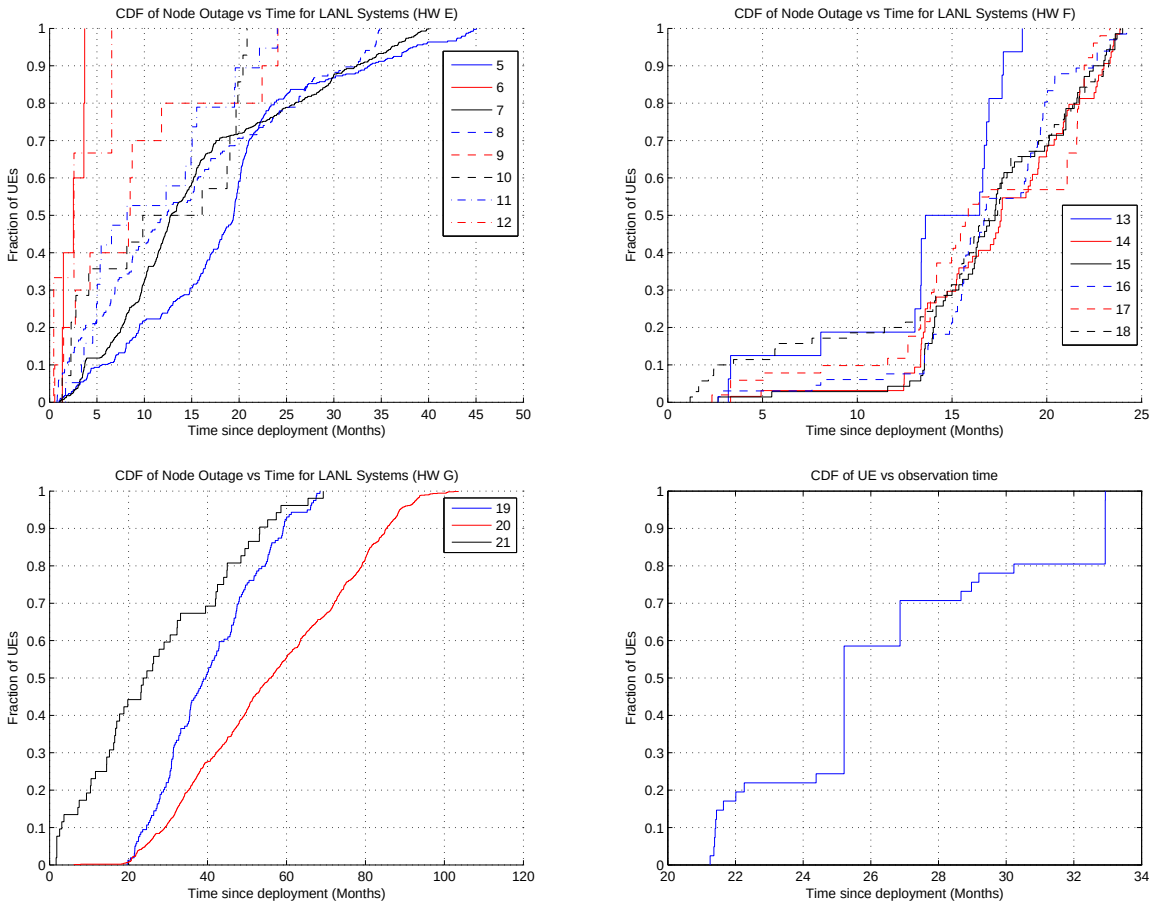

Figure 2: Distribution of node outages over time for LANL and Argonne systems. (Top left) Systems with hardware type E. For some systems with more data there is a slight rate-decrease after 22 months. (Top right) Systems with hardware type F. There is a clear trend of increased failure rate after 13 months. (Bottom left) Systems with hardware type G which is older. Failure rate over a long time is relatively steady. (Bottom right) Node UE rate for BG/P vs time, limited to the observations between month 21 to 33 since system deployment. The overall trend is steady.

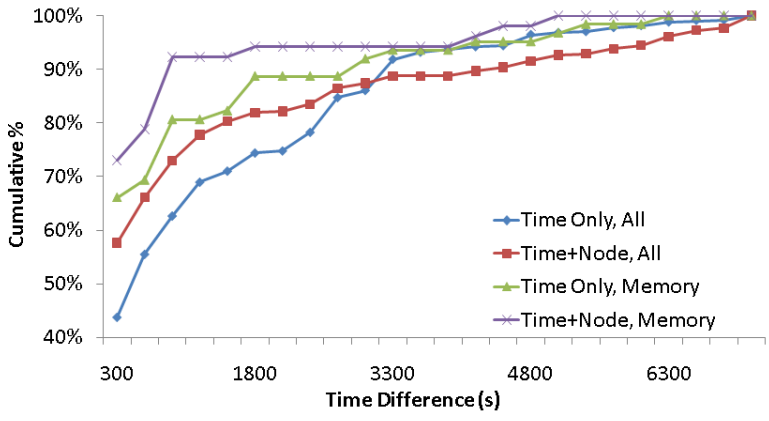

Figure 3: Results of matching job failed status with node outage for LANL system 5: CDF of match results vs the time difference. This matching was done using a very long window. The majority of memory-related errors happen within 5 minutes of node failure.

Table 3: CPU and real time lost due to outages for different matching methods. Memory-related job failures account for at least 15% of lost computing time.

Match Method / Outage Type	CPU Time Total (s)	CPU Time Median (s)	Real Time × CPU Total (s)	Real Time × CPU Median (s)
Time Only, All	413,665,817	975	519,480,711	90,560
Time Only, Memory	74,569,673	452,434	78,828,852	540,160
Memory %	18%		15%	
Time+Node, All	353,121,227	212,504	435,003,699	495,280
Time+Node, Memory	74,389,162	677,726	78,585,148	876,864
Memory %	21%		18%	

matched memory events stay relatively the same. This supports the fact that the majority of memory-related node outages can be matched to a job failure in a very short period of time.

Based on the failed jobs that were matched to memory events, we estimated the computing time lost due to memory errors, assuming no checkpointing was done and the the entire job has to be started over. Table 3 shows the lost compute time due to memory errors. Calculation was done using both the reported CPU time and the real time. While job failures are rare in general (1,800 jobs failed out of 300,000 submissions), the amount of lost compute time can still be significant since these are typically long jobs running across a large number of nodes. Nevertheless, it is clear that memory errors are responsible for at least 15% of the total lost compute time for system 5. Reducing the amount of memory errors can have an impact on job throughput.

5 Correctable Errors

After studying the characteristics of the uncorrectable errors, we turn our attention to the correctable errors. We look at temporal and spacial correlations between correctable errors. Understanding correlations of CEs is important in practice since it is more difficult for ECC algorithms to correct them if they are correlated in space or time. Correlation between correctable errors may also be indicators of hardware failures (rather than soft errors). While they do not immediately lead to machine crash or shutdown, repeated occurrences of correctable errors may increase the access time of DRAM (since ECC needs to be invoked for every access) or develop into uncorrectable errors in the future. We seek to understand the likelihood of CEs developing into UEs, and try to offer some insight to the nature of the error (hard/soft). We also wish to gain knowledge that would perhaps allow us to predict errors in the future.

We obtained ANL BG/P logs which contain entries for correctable errors. Several types of CE instances were recorded: single symbol, double symbol and chipkill CEs, where a symbol is 3 bits wide. Additional fields were collected to help identify the origin of the error: the error node, chip, and physical address. The first instance of a CE during execution of a job is reported along with the corresponding chip and address information. However, further CEs on the same location are not reported for the remainder of the job. At the end of each job, a summary message is reported stating the total number of CEs for each type that occurred. The summary messages, which are capped at 65,535, provide a more realistic estimation of the number of errors on a node than the first instance reports. For analyzing distributions at the chip level, we use the first report instead since the summary messages do not contain individual chip information. Therefore the total number of errors identified at the chip level is smaller than that at the node level. However, we believe that this does not significantly alter the trends in the distributions.

In the remainder of this section, we discuss the characteristics of correctable errors based on their distribution over time, across the error nodes and DRAM chips within nodes.

5.1 Summary Characteristics

The severity of correctable errors depends on the number of bits that required correction. In the most common case the error bits are contained within one *symbol*, or the unit for computing the ECC syndrome. More severe errors involve more ECC symbols and happen less often. The BG/P is a specially designed system and supports correction of single- and double-symbol errors, where a symbol is 3 bits wide. The bits on the memory bus are also arranged such that the failure of an entire DRAM chip is correctable - *Chipkill* correctable error. One reason for having strong ECC capabilities in the system is the difficulty in replacement. The DRAM chips are soldered directly on the node card and cannot be replaced without swapping the entire node card.

First we present some high level statistics for CEs in the BG/P system. One measure of reliability of de-

Table 4: Summary of correctable errors statistics during observation period

Error Type	Total Errors	FIT /MBit	Error /MBit /Yr	Node MTBF (Hrs)	Chip MTBF (Hrs)	Nodes Affected (/Yr %)	Mean Errors /Node	Median Errors /Node
Single	383,530,603	51,210	0.46	0.95	38	868 (2.12%)	441,855	16
Double	167,611,259	22,380	0.20	2.18	87	438 (1.07%)	382,674	40
Chipkill	82,762,974	11,051	0.10	4.42	177	404 (0.99%)	204,858	45
All	633,904,836	84,641	0.76	0.58	23	999 (2.44%)	634,539	26

vices is the mean time between failures (MTBF). Often the mean time to the first failure (MTTF) is also of interest, but since we did not observe the BG/P system from the beginning of its lifetime, it is impossible to know if errors happened prior to the start of our observation.

Memory errors can also be measured by Failures in Time (FIT), which is the number of failures per billion device hours per MBit. The FIT values reported in literature range from hundreds to tens of thousands FIT per Mbit. A source of discrepancy between the reported numbers could be due to inclusion of hard or soft errors in the particular measurements [23, 24, 34]. The correctable errors present in our logs could be either hard or soft errors as the system has no way to distinguish the root cause.

Table 4 lists the CE statistics for different error types at the node level. Single symbol errors are the most dominate CEs on nodes with 383,530,603 occurrences, and the event counts decrease for stronger ECC. Looking at the annual error rate per MBit and nodes affected and comparing to similar measures in Table 1, we can see that the incident rate of CEs is much higher than UEs.

The FIT values we obtained are comparable to those found in [34] which was also a study on large-scale production systems, but different to some others [23, 24] which were smaller or laboratory studies. This translates to a fairly low MTBF at the node level, but slightly higher at the chip level since there are 40 DRAM chips per node. Furthermore, the logs we collected capped the count of CEs at 65,535, therefore the totals presented here may be an underestimation. Lastly, it is important to point out each ECC symbol in the BG/P system is 3 bits wide, and thus a single symbol may in fact contain multi-bit errors.

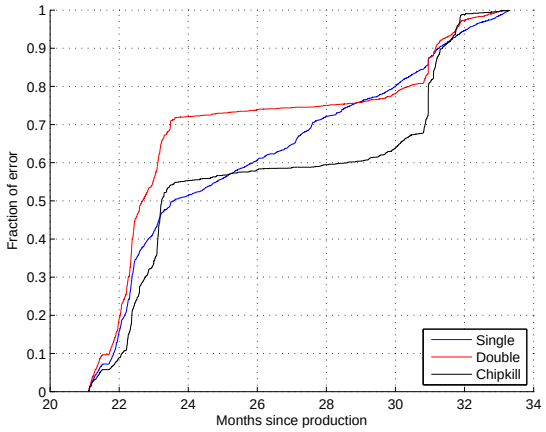

Figure 4: CE distribution for different error types over the observation time period.

5.2 Temporal Characteristics

While the FIT and MTBF numbers provide an overview of the CEs that happen in the system, we would like to understand how these errors are distributed in time and see if there are trends to when the errors happen, perhaps allowing us to predict errors based on job run time or system age. Figure 4 plots the CDF of errors over the observation period. The x-axis denotes the months relative to system production time. We can see that error rates for different CE types are in fact not uniform. The error rates for all CEs begin to level off around month 23. The rate for single symbol errors is steady thereafter, but double symbol and Chipkill CE rates begin to climb again after 31 months before flattening out at month 32. This suggests clustering of errors of these two types. Another possible explanation is that jobs were running for long periods of time and reported their CE counts near each other, but this is hard to tell without assembling the complete job history. It is also important to point out that do not know the error rates for the system before the start of our observation.

5.3 Error Distribution Across Nodes

While we see that the distribution of CEs is not uniform in time, it is also important to know how they are physically distributed on the hardware. This can help us design better prediction or mitigation techniques using any spacial locality that may exist. We first look at how CEs are distributed across the error nodes, then look at the chips within each node.

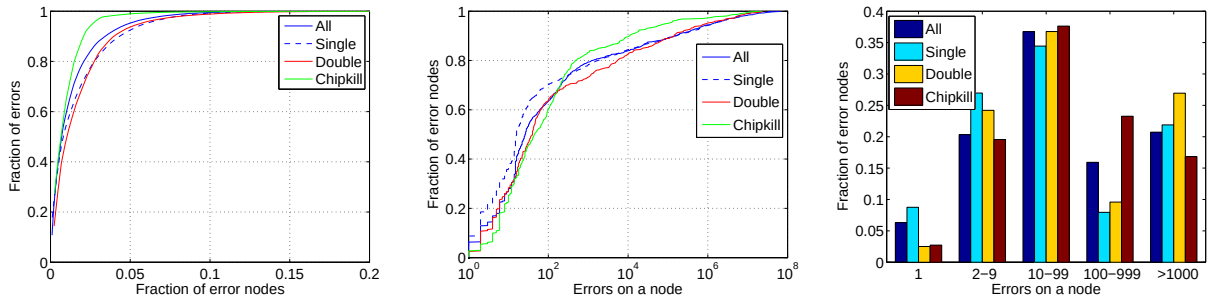

Figure 5: Error distributions with respect to error nodes (Left) Distribution of errors across error nodes - x-axis is fraction of error nodes for each type, y-axis is cumulative fraction of all errors of that type (Centre) CDF of error frequencies of error nodes - x-axis is number of errors of a certain type on a single node, y-axis is cumulative fraction of all error nodes of that type. i.e. *P(node has $\leq$ X number of errors)* (Right) Distribution of error frequencies per error node

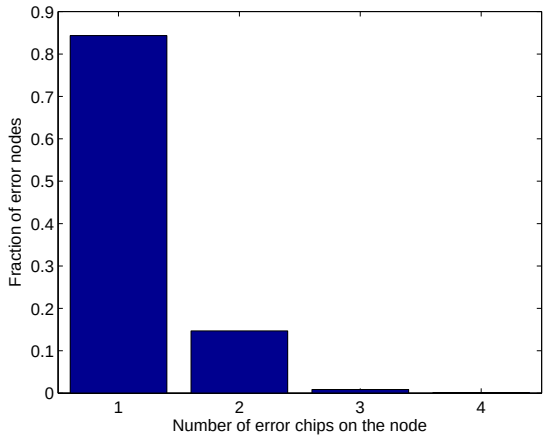

Figure 6: Distribution of number of error chips per error node. Over 80% of the chips only have one error chip.

Table 4 gives the number of error nodes for each CE type, as well as the mean and median errors per error node. The number of error nodes is very small compared to the total number of nodes in the system. The high mean and low median suggest a rather uneven distribution of CEs between the error nodes.

Figure 5 presents the distribution of CEs with respect to the error nodes of corresponding error types. The leftmost graph plots the cumulative fraction of CEs against the fraction of error nodes. For example, the point (x, y) represents that x fraction of error nodes account for total y fraction of errors. For all the error types, we find that over 90% of the errors are located in 5% of the nodes. The heavy concentration suggests that these nodes may in fact have hardware failures.

The concentration of errors is further confirmed by the center graph in Figure 5, where we plot the CDF of node error counts. The top 20% of error nodes of each type contains hundreds to millions of errors. The node histogram for different ranges of error counts is shown in the rightmost plot of Figure 5. For all CE types the most common case is between 10 and 99 errors on the same node.

The distribution of errors within nodes suggest that errors are highly localized and a small portion of nodes are responsible for a large number of CEs. This leads us to believe that the errors are caused by faulty hardware rather than random external events.

5.4 Error Distribution Across Chips

To investigate the exact location of errors within a node, we extracted information about the chip where the error occurred. The summary messages at the end of each job does not contain location information for each error. We resort to the first report of each address during job execution. This causes us to underestimate the total number of errors that occurred, however, we can still identify repetition on the same address across jobs.

Figure 6 shows the breakdown of nodes according to the number of error chips. The plot clearly indicates that over 80% of error nodes only contain a single error chip. Further confirming that errors are highly localized at the chip level. What we see as an error node is often a single error chip within that node.

After discovering that the errors are highly localized, we focused on how errors are distributed across chips. Table 5 lists the error characteristics at the chip level. As expected, the number of error chips is very close to the number of error nodes. The number of chips with double symbol errors is slightly higher due to the fact that the double symbol error may be across two chips. Chipkill errors are not included here because the algorithm can only identify two potential error chips without certainty.

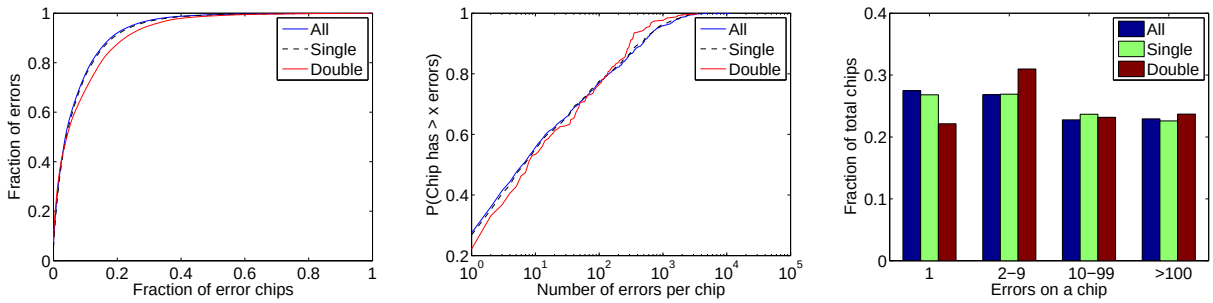

Figure 7: Error distributions with respect to individual error chips (Left) CDF of errors across chips - x-axis is fraction of error chips for each type, y-axis is cumulative fraction of all errors of that type (Centre) CDF of error frequencies on error chips - x-axis is number of errors of a certain type on a single chip, y-axis is cumulative fraction of all error chips of that type. i.e. *P(chip has* $\leq$ *X number of errors)* (Right) Distribution of error frequencies per error chip

Table 5: Summary of error chip characteristics, Chipkill errors are excluded since it is uncertain which of 2 possible chips generated the error

Error Type	Error Chips	Mean	Median
Single	1,018	154	7
Double	384	64	8
Either	1,208	151	7

The mean and median values are also very close to nodes with the same type of CE. This suggests that the error distribution across chips is similar to that of nodes.

We plotted the CE distribution over the population of error chips using the same methods in Figure 5. The three graphs in Figure 7 confirms that the distribution of errors across chips is has the same property to the distribution across nodes. The left plot of Figure 7 shows that over 80% of the errors (one type or overall) originates from 20% of chips. The centre plot of per chip error counts also closely resembles the case for nodes, with the hundreds and thousands of errors originating from the top 20% of error chips. Lastly, the right plot showing the fraction of chips in each tier of error count is roughly even over the log-scaled categories, slightly different than those on nodes. The localization of errors is not as extreme at the chip level because the log messages recorded less error counts than that at the node level. However, the property of the distributions remain similar.

The CE distributions at the node and chip level yielded interesting results. The reported errors originate from a small percentage of nodes and chip. Consequently, these devices also have high number of errors. We believe hardware failure is responsible for extremely large number of recurring errors, and investigate these errors within chips further in Section 6.

5.5 Degradation

The BG/P RAS logs contain different types of correctable errors as well as uncorrectable errors. In addition to looking at these error classes individually, we can also analyze the progression from one type of error to another. It would be useful for system administrators to identify potentially faulty hardware before uncorrectable errors happen.

To determine the accurate timestamp of messages, we used the first instance report of errors during job execution. For any particular error, we look for degradation by searching for another error on the same node within a specified time window. Table 6 lists the breakdown of different degradation scenarios for various time windows. In particular, we studied the degradation scenarios where more severe errors develop after encountering less severe errors on the same node. For example, 22.66% of single symbol errors are followed by at least one double symbol error within one day.

From Table 6, we can see that single symbol errors have a good probability of eventually developing into double symbol errors on the same node, ranging from 22.66% to 44.86% depending on the time window. The probability of single symbol errors further developing to more severe errors is much lower, although not to be disregarded due to the large number of errors that happen in the system.

Degradation also exists from double symbol errors. The probability of developing Chipkill errors after double symbol errors increases from 4.66% to 25.78% for increasing time windows. Double symbol errors also have a higher likelihood of developing into UEs. This may be because double symbol errors can be spread across two chips, which may each develop further errors and become uncorrectable even by the Chipkill algorithm.

An interesting fact is that we found no cases of Chipkill errors leading to uncorrectable errors. This may be

Table 6: Probability of different error degradation scenarios at different time windows

Time Window (days)	1	7	14	30	60	90	∞
Single symbol to double symbol	22.66%	32.03%	36.41%	39.76%	41.78%	42.67%	44.86%
Single symbol to Chipkill	0.57%	2.09%	2.90%	3.42%	3.75%	3.84%	4.19%
Single symbol to UE	0.02%	0.03%	0.03%	0.03%	0.03%	0.03%	0.04%
Double symbol to Chipkill	4.66%	15.99%	19.41%	23.63%	24.48%	24.69%	25.78%
Double symbol to UE	0.47%	0.60%	0.62%	0.62%	0.63%	0.63%	0.70%
Chipkill to UE	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Any CE to UE	0.08%	0.11%	0.11%	0.11%	0.11%	0.11%	0.13%

Table 7: Breakdown of chip error counts and unique error addresses. (Left) Breakdown of BG/P error chips according to number of errors on chip. (Right) Breakdown of chips with multiple errors by number of unique error addresses.

Chip Type	Count	Percent
Single error on chip	332	27.5%
Multiple errors on chip	876	72.5%
Total	1,208	100%

Chip Type	Count	Percent
Single unique error address	401	45.8%
Multiple unique error addresses	475	54.2%
Total	876	100%

due to the small number of nodes recorded for both cases, as the degradation to UE from other scenarios is also small. Nevertheless, an entire chip failure may not become any more severe until some other chip has an error.

For all the time windows, we see that one or two weeks is a usually a fair approximation of the probabilities of developing further errors. This also means that degradation is not likely to happen over very long periods of time and can be detected relatively quickly when it happens.

Overall, we observe that there is some degradation from one form of CE to a more severe form in a short time. However, the degradation to uncorrectable errors is much less likely. The is likely due to the fact that the strong ECC is implemented to correct errors from one source chip. The nodes with potentially more than one error chip (double symbol error), may be more susceptible to uncorrectable errors because the ECC hardware is less capable of correcting multi-chip errors.

6 Device Failure Modes

The analysis in Section 5 shows that errors are highly localized on the hardware. Recurring errors, whether correctable or not, is likely caused by faulty hardware and not external events. In this section we further analyze the location of the correctable errors in attempt to understand their root cause. To the best of our knowledge, this is the first study of failure addresses from large-scale production systems. We base our analysis on the incidents of single symbol and double symbol errors on error chips, which are the majority of observed CEs in the system. We also believe this is an underestimation of the total amount of errors since only the first incident on the same location during a job is recorded. Re-occurrences on the same address may happen more often.

6.1 Error Counts and Error Addresses

We first breakdown the error chips according to the number of errors they have. If only a single error occurs on a chip, it can be explained as a soft error caused by a random event from the environment, such as an alpha particle. However, multiple errors on the same chip can indicate hard or intermittent errors due to device failure. It is highly unlikely that alpha particles struck the same chip twice over the course of one year. In this case, we look to the address information to suggest the potential failure mode. It is obvious that a chip with a single observed error will only have one error address. For chips with multiple errors, the errors can be repeating on the same address, which points towards a potential defect in the corresponding DRAM cell. Alternatively, if multiple errors on the chip are spread across different addresses, we would like to see if the physical faulting locations are related, which would mean some clustering of failed cells. To the best of our knowledge, this is the first work with location information from large-scale production systems.

The left half of Table 7 shows the number of chips with single and multiple errors. Approximately a quarter of the error chips only have a single error, which may be caused by soft errors. The remaining 70% of the error chips have multiple error addresses. The right side of Table 7 shows that for these chips, nearly half only have one error address (slightly over a third of all error chips). The single error address strongly suggests that these are

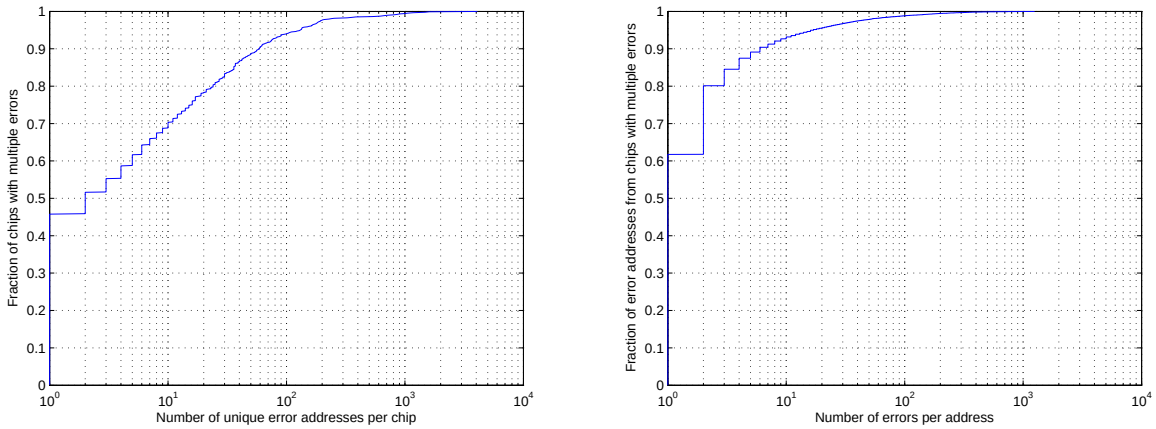

Figure 8: Error distributions for chips with multiple errors (Left) CDF of number of unique error addresses on for chips with multiple errors. 46% of chips only have one error address, and 90% of chips have less than 100 error addresses. (Right) Distribution of errors counts per address on chips with multiple errors. Over 90% of the error addresses on these chips have < 10 errors.

hard errors and the chip is in fact defective. The second category of chips with multiple errors have multiple error addresses. We have to look at the distribution of error addresses to better understand the failure mode.

Figure 8 illustrates the distribution of error addresses on chips. The left plot shows the CDF of number of unique error addresses the chips with multiple errors. As expected, 46% of these chips only have a single address. Even though the rest of the chips have multiple error addresses, we observe that overall the majority of chips with multiple errors have less than 100 faulting address, with the a small number of chips having hundreds of addresses.

The distribution of error counts per address is presented in the right plot in Figure 8. The addresses collected from chips with multiple errors are organized by their error counts. The majority of error addresses only have less than 10 errors, but some addresses may have hundreds of errors. The disparity in number of faulting addresses and error counts motivates us to further look at the physical layout of the error chips.

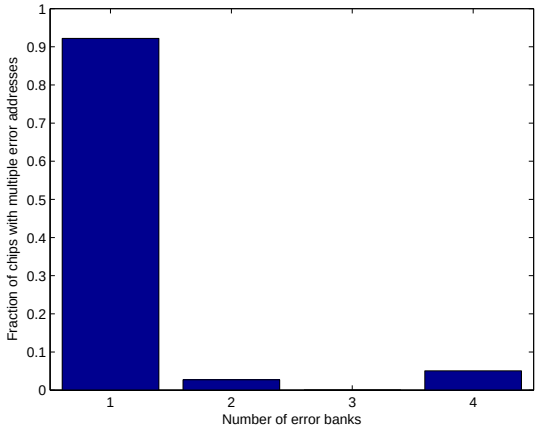

Figure 9: Distribution of unique error banks on chips with multiple error addresses

6.2 Rows, Columns, and Banks

DRAM cells inside a chips are laid out in two-dimensional arrays. Many such arrays are layered on top of each other. To address a single cell, the appropriate layer, or *bank* must specified, then the *row* and *column* is specified to index a cell within the bank.

Previously we looked at only at the distribution of errors with respect to the memory addresses. It would be interesting to show how these relate to physical locations on the DRAM chip such as row, column or bank. Reverting memory addresses to their physical locations is non-trivial because this mapping is different for every memory controller. We obtained information from Blue Gene architects and performed the reverse-mapping using the memory controller specification for IBM Blue Gene/L (BG/L) systems [16]. The memory controller specifications for BG/P and BG/L are believed to be similar.

We first check the distribution of addresses over banks for chips that contain more than one error address. Figure 9 shows the number of banks for chips with multiple error addresses. Out of 4 banks available from each DRAM device, it is clear that most chips with multiple errors are confined to one bank.

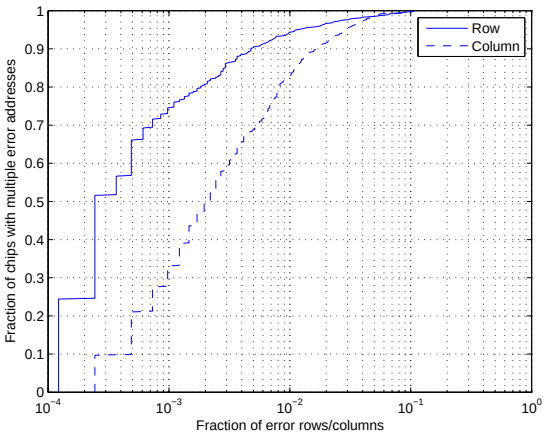

Figure 10: CDF of unique error rows and columns on chips with multiple error locations, scaled by available rows and columns on the chip

Within each bank, the errors are also localized. For each chip with multiple error addresses, we calculated the number of rows and columns spanned by the error addresses, normalized by the total available rows and columns within a chip. Figure 10 plots the fraction of rows and columns that contains errors. There is a tendency for errors on the same chip to appear in a small fraction of rows. The spread over columns is around one order of magnitude larger for the majority of error chips. This indicates that the multiple errors we observe tend to be on the same row, meaning that these rows are likely to be defective.

7 Conclusion and Future Work

Memory devices are integral in any computer system. Its capabilities are often key factors in the systems' overall performance. Unfortunately, it is also one of the most frequently replaced hardware components. To understand the characteristics of DRAM reliability, we analyzed data from three different sources: Los Alamos National Lab, Argonne National Lab, and SciNet Consortium.

Using node outage and replacement data, combined with reports of uncorrected errors and machine check exceptions, we computed the nodes affected by memory errors. In typical commodity systems, up to 10% of nodes are expected to have uncorrectable errors annually. The UE per MB per year is in the order of 10^{-5} to 10^{-6}. For some systems we found evidence of aging, and the behaviours of systems within the same hardware platform tend to be similar.From matching node outages to job failure logs, we estimate that up to 15% of total lost compute time is due to memory-related outages.

For correctable errors, the FIT value we measured from the Blue Gene/P system is around 85,000, which is much higher than some previous studies . Furthermore, the logs point out that correctable errors are very localized in production systems. 90% of the CEs happen on only 5% of the error nodes, and the source of the errors is almost certainly a single chip within the node. The hardware with the most errors can contain up to millions of errors in the single year of being observed.

There is also evidence that some error nodes develop more severe errors after observing less severe CEs. For instance, 40% of single symbol errors eventually develop into double symbol errors. Although it is very rare for CEs to develop into UEs. Furthermore, the majority of degradation can be observed in a short time window. The localization of errors and the degradation points to hardware errors as the major error source rather than soft errors.

Finally, we studied the error address on the chips to gain insight the nature of the error that has occurred. Approximately 27% of the chips only contain a single error, which might be an incidence of soft error due to alpha particles. Interestingly, 33% of the error chips have recurring errors in the same address, which is an indication for hard or intermittent errors.

When studying the distribution of physical error locations within a chip, we saw errors being confined to fewer number of rows than columns, and limited to one single error bank. The localization suggests that they are also likely clustered error due to hardware.

The observation of hard errors as a major failure mode motivates a wide range of future work. We would like to further examine the locality of errors. We hope to come up with models to explain and predict the different failure modes such as rows or columns. Based on these models, we would also like to be able to predict errors in production systems, which may allow graceful service degradation or shutdown. Furthermore, the characteristics of errors enables us to construct better mitigating mechanisms, both in hardware and at the operating system level. For example, we can enhance page isolation policies by incorporating the error models from field data.

8 Acknowledgments

I would like to thank my supervisor, Bianca Schroeder, and second reader Angela Demke Brown, for their encouragements and comments on early drafts of this study.

References

[1] Argonne Leadership Computing Facility. http://www.alcf.anl.gov/.

[2] Kernel Parameters, Linux Kernel Documentation. http://www.kernel.org/doc/Documentation/kernel-parameters.txt.

[3] Operational Data to Support and Enable Computer Science Research, Los Alamos National Laboratory. http://institute.lanl.gov/data/fdata/.

[4] SciNet. http://www.scinet.utoronto.ca/.

[5] Memory Hotplug, Linux Kernel Documentation. http://www.kernel.org/doc/Documentation/memory-hotplug.txt, 2007.

[6] *Intel Xeon Processor 5500 Series Datasheet Volume 2*, 2009.

[7] ASHBURN, J. HWPOISON. http://lwn.net/Articles/348886/, 2009.

[8] BAIRAVASUNDARAM, L. N., ARPACI-DUSSEAU, A. C., ARPACI-DUSSEAU, R. H., GOODSON, G. R., AND SCHROEDER, B. An analysis of data corruption in the storage stack. *Trans. Storage 4*, 3 (November 2008), 8–1.

[9] BAIRAVASUNDARAM, L. N., GOODSON, G. R., PASUPATHY, S., AND SCHINDLER, J. An analysis of latent sector errors in disk drives. In *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems* (New York, NY, USA, 2007), SIGMETRICS 07, ACM, pp. 289–300.

[10] BARTON, J. H., CZECK, E. W., SEGALL, Z. Z., AND SIEWIOREK, D. P. Fault Injection Experiments Using FIAT. *IEEE Transactions on Computers 39* (1990), 575–582.

[11] BRADY, C. Memtest86. http://www.memtest86.com/.

[12] CONSTANTINESCU, C. Impact of Intermittent Faults on Nanocomputing Devices. In *DSN 2007 Workshop on Dependable and Secure Nanocomputing* (2007).

[13] DAVID, F. M., AND CAMPBELL, R. H. Building a Self-Healing Operating System. In *Dependable, Autonomic and Secure Computing, 2007. DASC 2007. Third IEEE International Symposium on* (sept. 2007), pp. 3–10.

[14] DELL, T. J. A white paper on the benefits of chipkill-correct ECC for PC server main memory. *IBM Microelectronics Division Whitepaper* (1997).

[15] DU, Y., YU, H., JIANG, Y., DONG, Y., AND ZHENG, W. A Rising Tide Lifts All Boats: How Memory Error Prediction and Prevention Can Help with Virtualized System Longevity. *Sixth Workshop on Hot Topics in System Dependability* (2010).

[16] GOODING, T. personal communication, 2010.

[17] HAQUE, I. S., AND PANDE, V. S. Hard Data on Soft Errors: A Large-Scale Assessment of Real-World Error Rates in GPGPU. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on* (may 2010), pp. 691–696.

[18] HORST, R. W. Reliability Improvement Through Static RAM Sparing. Tech. rep., Tandem Computers, Inc., 1987.

[19] IBM JOURNAL OF RESEARCH AND DEVELOPMENT STAFF. Overview of the IBM Blue Gene/P project. *IBM J. Res. Dev. 52*, 1/2 (January 2008), 199–220.

[20] JACOB, B., NG, S. W., AND WANG, D. T. *Memory systems: cache, DRAM, disk*. Morgan Kaufmann Pub, 2007.

[21] KIM, J., SMOLENS, J. C., FALSAFI, B., AND HOE, J. C. PAI: A Lightweight Mechanism for Single-Node Memory Recovery in DSM Servers. *Pacific Rim International Symposium on Dependable Computing, IEEE 0* (2007), 298–305.

[22] LI, Q., AND PATEL, U. Enabling Memory Reliability, Availability, and Serviceability Features on Dell PowerEdge Servers, 2005.

[23] LI, X., HUANG, M. C., SHEN, K., AND CHU, L. An empirical study of memory hardware errors in a server farm. In *Proceedings of the 3rd workshop on on Hot Topics in System Dependability* (Berkeley, CA, USA, 2007), USENIX Association.

[24] LI, X., HUANG, M. C., SHEN, K., AND CHU, L. A realistic evaluation of memory hardware errors and software system susceptibility. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference* (Berkeley, CA, USA, 2010), USENIXATC10, USENIX Association, pp. 6–6.

[25] LI, X., SHEN, K., HUANG, M. C., AND CHU, L. A memory soft error measurement on production systems. In *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference* (Berkeley, CA, USA, 2007), USENIX Association, pp. 21–1.

[26] MAY, T. C., AND WOODS, M. H. A New Physical Mechanism for Soft Errors in Dynamic Memories. In *Reliability Physics Symposium, 1978. 16th Annual* (april 1978), pp. 33–40.

[27] OLINER, A., AND STEARLEY, J. What Supercomputers Say: A Study of Five System Logs. *Dependable Systems and Networks, International Conference on 0* (2007), 575–584.

[28] OPPENHEIMER, D., GANAPATHI, A., AND PATTERSON, D. A. Why do internet services fail, and what can be done about it? In *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems - Volume 4* (Berkeley, CA, USA, 2003), USITS03, USENIX Association, pp. 1–1.

[29] PINHEIRO, E., WEBER, W. D., AND BARROSO, L. A. Failure trends in a large disk drive population.

[30] SANDERS, N. J., AND MENDERICO, R. M. stressapptest. http://code.google.com/p/stressapptest/.

[31] SCHMOIGL, N. BadMEM-HOWTO. http://badmem.sourceforge.net/docu/BadMEM-HOWTO.html.

[32] SCHOPP, J. H., FRASER, K., AND SILBERMANN, M. J. Resizing Memory With Balloons and Hotplug. In *Linux Symposium* (2006), p. 305.

[33] SCHROEDER, B., AND GIBSON, G. A. A large-scale study of failures in high-performance computing systems. *Dependable Systems and Networks, International Conference on 0* (2006), 249–258.

[34] SCHROEDER, B., PINHEIRO, E., AND WEBER, W.-D. DRAM errors in the wild: a large-scale field study. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems* (New York, NY, USA, 2009), SIGMETRICS 09, ACM, pp. 193–204.

[35] SIEH, V., AND BUCHACKER, K. UMLinux - A Versatile SWIFI Tool. In *Proceedings of the 4th European Dependable Computing Conference on Dependable Computing* (London, UK, 2002), EDCC-4, Springer-Verlag, pp. 159–171.

[36] STOTT, D. T., FLOERING, B., BURKE, D., KALBARCZPK, Z., AND IYER, R. K. NFTAPE: a framework for assessing dependability in distributed systems with lightweight fault injectors. In *Computer Performance and Dependability Symposium, 2000. IPDS 2000. Proceedings. IEEE International* (2000), pp. 91–100.

[37] TANG, D., CARRUTHERS, P., TOTARI, Z., AND SHAPIRO, M. W. Assessment of the Effect of Memory Page Retirement on System RAS Against Hardware Faults. In *Dependable Systems and Networks, 2006. DSN 2006. International Conference on* (june 2006), pp. 365–370.

[38] THOMPSON, D., AND CHEHAB, M. C. EDAC - Error Detection And Correction, Linux Kernel Documentation. http://www.kernel.org/doc/Documentation/edac.txt, 2009.

[39] WALDSPURGER, C. A. Memory resource management in VMware ESX server. *SIGOPS Oper. Syst. Rev. 36*, SI (December 2002), 181–194.

[40] YIM, K. S., KALBARCZYK, Z., AND IYER, R. K. Measurement-based analysis of fault and error sensitivities of dynamic memory. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on* (282010-july1 2010), pp. 431–436.

[41] YOON, D. H., AND EREZ, M. Virtualized and flexible ECC for main memory. *SIGPLAN Not. 45*, 3 (March 2010), 397–408.